REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 27-11-2012	**2. REPORT TYPE** SBIR Phase 2 Year 1 Final Report	**3. DATES COVERED** *(From - To)* 28-11-2011 To 27-11-2012

4. TITLE AND SUBTITLE Open, Cross Platform Chemistry Application Unifying Structure Manipulation, External Tools, Databases and Visualization	**5a. CONTRACT NUMBER** W912HZ-12-C-0005
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Dr. Marcus D. Hanwell	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Kitware, Inc. 28 Corporate Drive, #204 Clifton Park, NY 12065-8662	**8. PERFORMING ORGANIZATION REPORT NUMBER** Topic #A10-110 Proposal #A2-4714
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) US Army Engineering Research & Development Center	**10. SPONSOR/MONITOR'S ACRONYM(S)** ERDC
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRUBUTION STATEMENT A: Distribution is approved for public release; Distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Report developed under SBIR contract for topic #A10-110. The overarching goal of this project is the creation of the leading computational chemistry workbench, making the premier computational chemistry codes and databases easily accessible to chemistry practitioners. This will be accomplished by creating an open, extensible application framework that puts computational tools, data, and domain-specific knowledge at the fingertips of chemists. A data-centric approach to chemistry, storing all data in a searchable database, will empower users to efficiently collaborate, innovate, and push the frontiers of research. This report documents the project at its mid-point, documenting an open and extensible approach to computational chemistry. Desktop integration with high performance computing resources, seamless execution of external computational chemistry packages from within a molecular editor. Next generation, scalable rendering techniques enable researchers to build and visualize very large simulations with data structures poised to enable client-server desktop functionality in the second half of the project. Integration of these features with powerful analysis tools and an informatics approach leveraging best-of-breed NoSQL databases, in order to store, search and retrieve relevant results when they are needed.

15. SUBJECT TERMS

Computational chemistry, cheminformatics, HPC, quantum chemistry, cross platform, open source, visualization, analysis, databases.

16. SECURITY CLASSIFICATION OF: U			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Marcus D. Hanwell
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	22 Including SF-298	**19b. TELEPHONE NUMBER** *(include area code)* (518) 371-3971 ext. 520

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

28 Corporate Drive
Clifton Park, NY 12065
518.371.3971

SBIR Phase 2 Year 1 Final Report

Contract Number: W912HZ-12-C-0005
SBIR Topic Number: A10-110

Open, Cross Platform Chemistry Application
Unifying Structure Manipulation, External Tools, Databases and Visualization

November 27, 2012

Principal Investigator: Dr. Marcus D. Hanwell
marcus.hanwell@kitware.com
Phone: (518) 371-3971 ext. 520
Fax: (518) 371-4573

Kitware Inc.
28 Corporate Drive
Clifton Park, NY 12065

Sponsored By:
U.S. Army
Engineering Research & Development Center (ERDC)

UNCLASSIFIED

Open, Cross Platform Chemistry Application

Unifying Structure Manipulation, External Tools, Databases and Visualization

Phase II SBIR Year 1 Report for Topic A10-110
Proposal Number A2-4714

Principal Investigator
Dr. Marcus D. Hanwell

Kitware, Inc.
28 Corporate Drive
Clifton Park, NY 12065
http://www.kitware.com/
518-371-3971

Contents

1 Technical Objectives

The overarching goal of this project is the creation of the leading computational chemistry workbench, making the premier computational chemistry codes and databases easily accessible to chemistry practitioners. This will be accomplished by creating an open, extensible application framework that puts computational tools, data, and domain-specific knowledge at the fingertips of chemists. A data-centric approach to chemistry, storing all data in a searchable database, will empower users to efficiently collaborate, innovate, and push the frontiers of research.

As the power of our computational resources grows, computational chemists face a growing discrepancy between our ability to run calculations/simulations and our ability to meaningfully store, search, retrieve and analyze data. As the sophistication of the computational codes grow and access to powerful computational resources becomes more commonplace, there is an increasingly steep learning curve to effectively using new computational tools and analyzing their output. Our objective is to make the lives of computational chemists easier by making these tools accessible to a wider range of chemists. The specific goals of the Phase II project are outlined below, and summarized in Figure 1.

- Develop an extensible, plugin-based, flexible chemistry application and library
- Develop an application for easily using HPC resources from desktop applications
- Develop a specialized desktop database application for chemical information
- Develop chemistry specific analysis and visualization techniques
- Develop a specialized file format capable of storing large data
- Develop a library for injesting and calculating electronic structure
- Develop a "chemistry workbench" offering state-of-the-art tools to the community

Kitware has been very successful for more than a decade by building collaborative innovation platforms that allow us to work with the best research groups in the world to leverage their research and development. This framework will position us to be able to pursue fruitful collaborations in chemistry and several other related areas.

2 Work Summary

This section summarizes the work done in the first year of a two-year Phase II project, with discussion of progress made thus far in achieving the overall goals of the project, as outlined in the proposal referenced. This project involves three major areas of development (shown in Figure 1), with several open source projects supporting the work (shown in Figure 2).

The projects shown in Figure 2 summarize those being developed or extended as part of this project, with an indication of application domain and type. There are three user facing graphical applications that are aimed at being used from the desktop to do research in the broad area of computational chemistry: Avogadro, ChemData, and MoleQueue. Each of

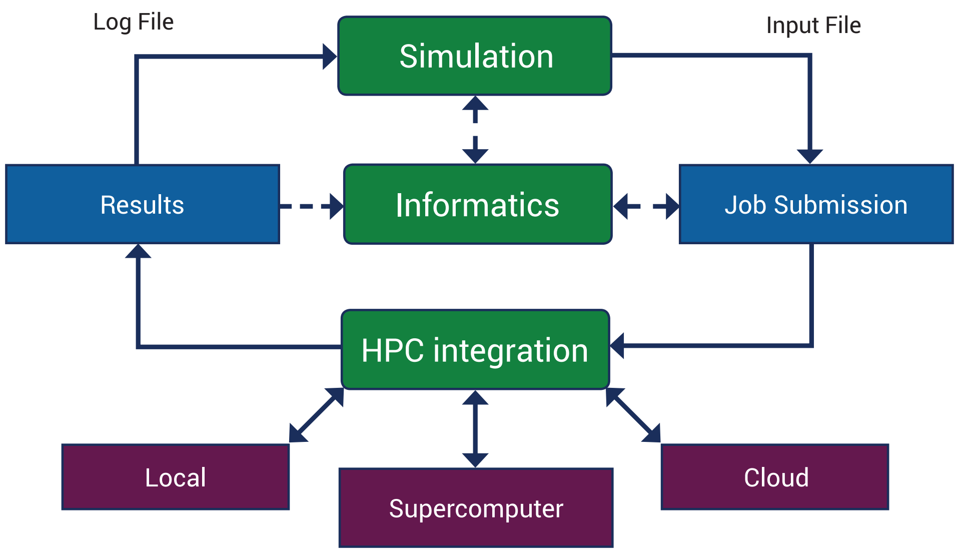


Figure 1: Open Chemistry workflow, with Open Chemistry applications filling the roles in green, and the flow of data indicated by arrows.

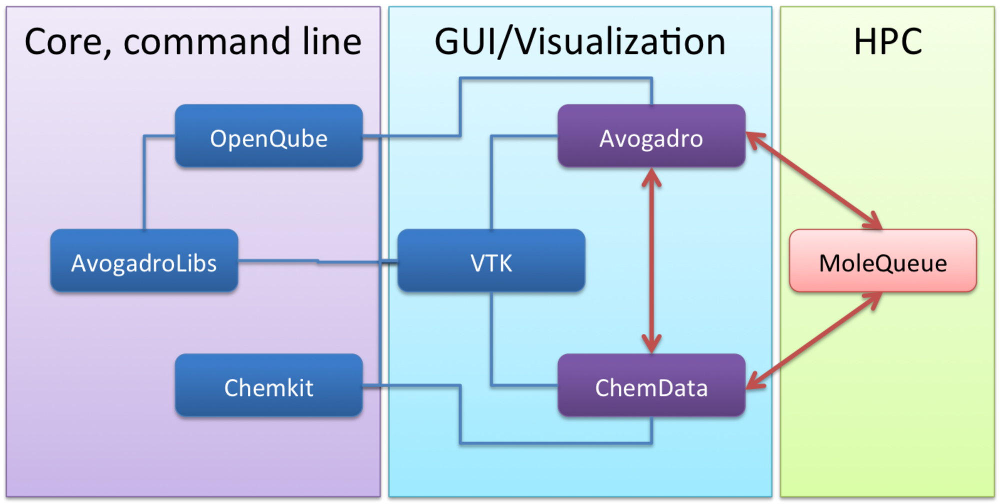


Figure 2: Open Chemistry projects grouped by basic dependency and application area.

these applications is specialized to deal with distinct domains, but designed to be used in unison with the other applications.

These projects build upon existing libraries where possible. All three are written in C++, make use of several cross-platform, open source libraries, such as Qt[1] and CMake,[2,3] in order to build on many platforms. In addition to the more generic libraries and tools, several specialized libraries are also being developed or extended in order to support the Open Chemistry work. The VTK project[4–6] is one of the oldest C++ visualization libraries still actively developed, and is one of Kitware's core projects. It has being augmented with additional chemical data structures and visualization types that complement the existing visualization approaches in order to make it a more compelling choice for chemists. It sits between the GUI/visualization libraries and the core/command line in where it can be deployed and used—featuring both data pipeline, CPU or GPU rendering and parallel techniques— for data analysis and reduction.

The Chemkit, AvogadroLibs and OpenQube components provide the majority of the chemistry-specific functionalities necessary, such as standard descriptors, file formats, force fields, data structures, algorithms, and post-processing calculations on computational chemistry output files. In addition to new functionality developed in these libraries, the Open Babel[7,8] and RDKit[9] libraries can also be used; for example, in the generation of 2D chemical structure depiction in batch mode and file format support/conversion.

2.1 Software Process and Project Dissemination

The Open Chemistry applications and libraries[10] are developed as independent projects grouped under the Open Chemistry project, with a community site at http://www.openchemistry.org/. Many open source projects use a somewhat standard software process,[11] which has been adopted in a slightly modified form for the Open Chemistry projects (Figure 3).

Several key resources have been put in place for the projects:

- Community website dedicated to Open Chemistry projects
- Git source code repositories (Kitware, mirrored to Github and Gitorious)
- Online code review tool (Gerrit)
- Online software quality dashboards (CDash)
- Community wiki pages (MediaWiki)
- Bug tracking and project management tools
- Mailing lists

The community website (Figure 4) acts as an entry point to the project, and gives a brief introduction to the projects with links to specific resources. The projects use non-reciprocal BSD license and distributed version control (Git) in order to enable customization of private branches and shared open branches. Git also gives us to ability to mirror the code base be be mirrored in multiple locations, with full access to the history and private mirrors possible within organizational units.

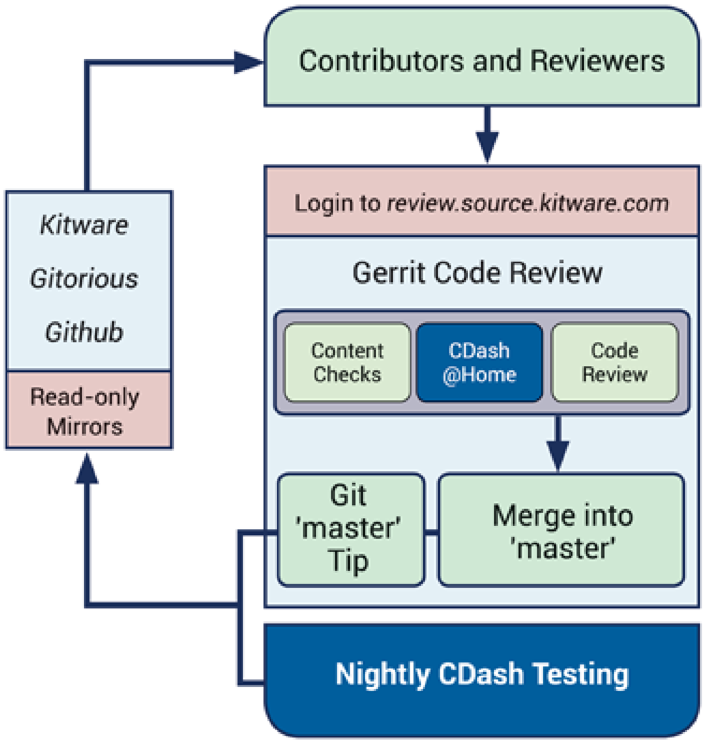
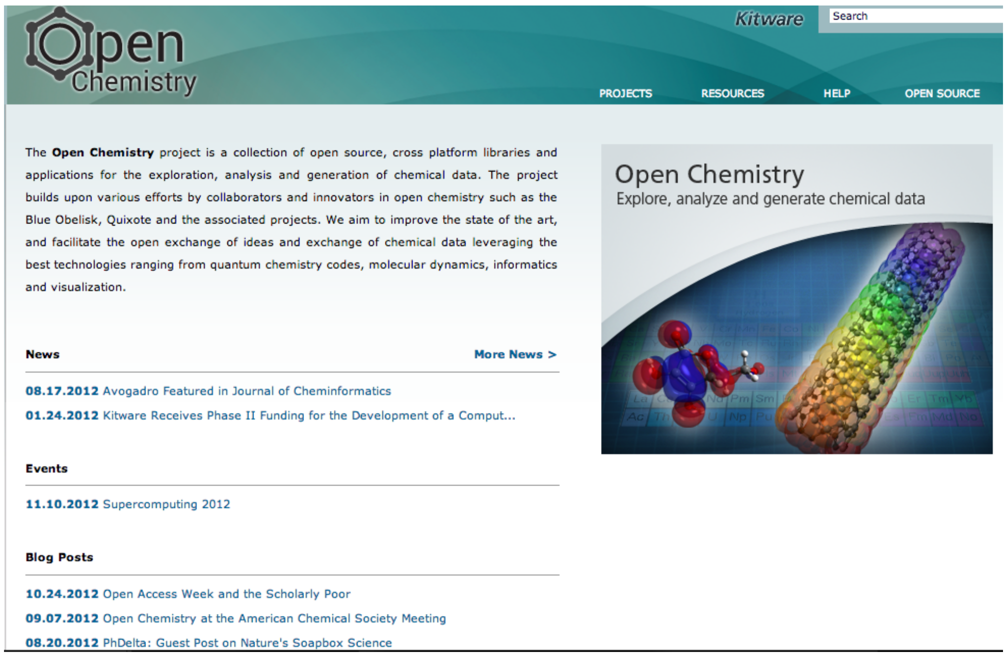


Figure 3: The software process used for Open Chemistry projects.



The **Open Chemistry** project is a collection of open source, cross platform libraries and applications for the exploration, analysis and generation of chemical data. The project builds upon various efforts by collaborators and innovators in open chemistry such as the Blue Obelisk, Quixote and the associated projects. We aim to improve the state of the art, and facilitate the open exchange of ideas and exchange of chemical data leveraging the best technologies ranging from quantum chemistry codes, molecular dynamics, informatics and visualization.

News **More News >**

08.17.2012 Avogadro Featured in Journal of Cheminformatics
01.24.2012 Kitware Receives Phase II Funding for the Development of a Comput...

Events

11.10.2012 Supercomputing 2012

Blog Posts

10.24.2012 Open Access Week and the Scholarly Poor
09.07.2012 Open Chemistry at the American Chemical Society Meeting
08.20.2012 PhDelta: Guest Post on Nature's Soapbox Science

Open Chemistry
Explore, analyze and generate chemical data

Figure 4: The Open Chemistry homepage.

The Gerrit code review system,[12] developed by Google as an open-source project for the Android operating system, enabling online review of code submissions from anyone while retaining control of what code is accepted into the code base. This has been combined with nightly software build testing on all three major platforms for merged code and testing of proposed changes using CDash@Home[13] (an open-source project developed at Kitware to address the need for testing arbitrary branches automatically). This level of automation gives Open Chemistry projects the ability to maintain high code quality, and reviewers are free to focus on verification of code correctness while the automated systems assure that portable code that works on all major platforms.

The projects are in several independent repositories which are then included in an Open Chemistry repository that is capable of building all of the projects and their major dependencies on the major platforms (Linux, Windows and Mac). The Open Chemistry repository provides an easy entry point for new developers, and using CDash with nightly testing provides binaries for Windows and Mac that are automatically generated every night. These are available both on the dashboard and on a site set up to help users find the appropriate binaries.

2.2 Data Models and Communication Strategies

Early on in the development of the project, the Javascript Object Notation (JSON) format[14] was settled upon for simple serialization/deserialization of data and inter-process communication. The JSON format is an extremely simple industry standard being increasingly used in places where formats such as XML were once used. It has the distinct advantage of being a very simple format that is possible to parse easily, and has support in a diverse array of programming languages from compiled languages such as C, C++ and Fortran through to Java, Python, Perl, and JavaScript.

At its core, the JSON data structure consists of key/value pairs and ordered lists. Both concepts are universal to most programming languages, enabling a great deal of freedom in language choice for data exchange. There are several C++ JSON libraries available, and two were chosen for use in the Open Chemistry project—JsonCpp which is a very small MIT licensed library using only STL, and Qt 5's JSON classes which were backported to Qt 4.8 to enable its use in Qt 4.8 and Qt 5.0 based projects. The Python language also has native support for JSON data structures using dictionaries, and JavaScript also has good support.

The MongoDB project[15] was chosen as a scalable NoSQL data store for the cheminformatics components of this work. The MongoDB project uses a binary form of JSON called BSON,[16] which follows many of the same principles as JSON, but stores data in raw binary and has optimized the data structures to support fast reading and writing of documents. This means that moving data from the backend data store to applications and over inter-process communication channels is a very easy process. Several libraries are also available with native BSON support, such as C, C++, and Python. BSON has the distinct advantage of IO speed and the ability to store raw binary data, such as PNG images, binary file fragments etc with binary data length encoded in the standard representation and support for most basic types.

The use of JSON and BSON in these projects also prompted the development of a JSON/BSON data model to represent chemical structures. This was developed to mirror many of the structures already developed for the Chemical Markup Language, and translation between the two formats should be lossless. They are both extensible formats building on widely-accepted industry standard data exchange formats. Neither is especially suited to very large data, but can be coupled with a separate binary data store to give semantically rich data documents that point to larger blobs of binary data when appropriate. An example of a small structure in Chemical JSON:

```
{
  "chemical json": 0,
  "name": "ethane",
  "inchi": "1/C2H6/c1-2/h1-2H3",
  "formula": "C 2 H 6",
  "atoms": {
    "elements": {
      "number": [  1,   6,   1,   1,   6,   1,   1,   1 ]
    },
    "coords": {
      "3d": [  1.185080, -0.003838,  0.987524,
               0.751621, -0.022441, -0.020839,
               1.166929,  0.833015, -0.569312,
               1.115519, -0.932892, -0.514525,
              -0.751587,  0.022496,  0.020891,
              -1.166882, -0.833372,  0.568699,
              -1.115691,  0.932608,  0.515082,
              -1.184988,  0.004424, -0.987522 ]
    }
  },
  "bonds": {
    "connections": {
      "index": [ 0, 1,
                 1, 2,
                 1, 3,
                 1, 4,
                 4, 5,
                 4, 6,
                 4, 7 ]
    },
    "order": [ 1, 1, 1, 1, 1, 1, 1 ]
  },
  "properties": {
    "molecular weight": 30.0690,
    "melting point": -172,
    "boiling point": -88
  }
}
```

The key names map well to the XML nodes in CML documents,[17] and this structure can easily be stored directly in MongoDB as a document (or object within a document) or passed between processes as JSON. Readers can check for the existence of known keys, and JSON documents can be built up by various subroutines using a simple in-memory model.

2.3 ChemData

The ChemData application is developed using C++, Qt, MongoDB and VTK, with some of its cheminformatics functionality coming from Open Babel and Chemkit. This application is focused on facilitating the deposition of chemical data in a scalable database, adding various properties to the molecules, and facilitating the dynamic visualization and analysis of aggregate data. It has been generalized to support connections to different MongoDB database servers, including the use of shared servers, in the cases where very large databases are required.

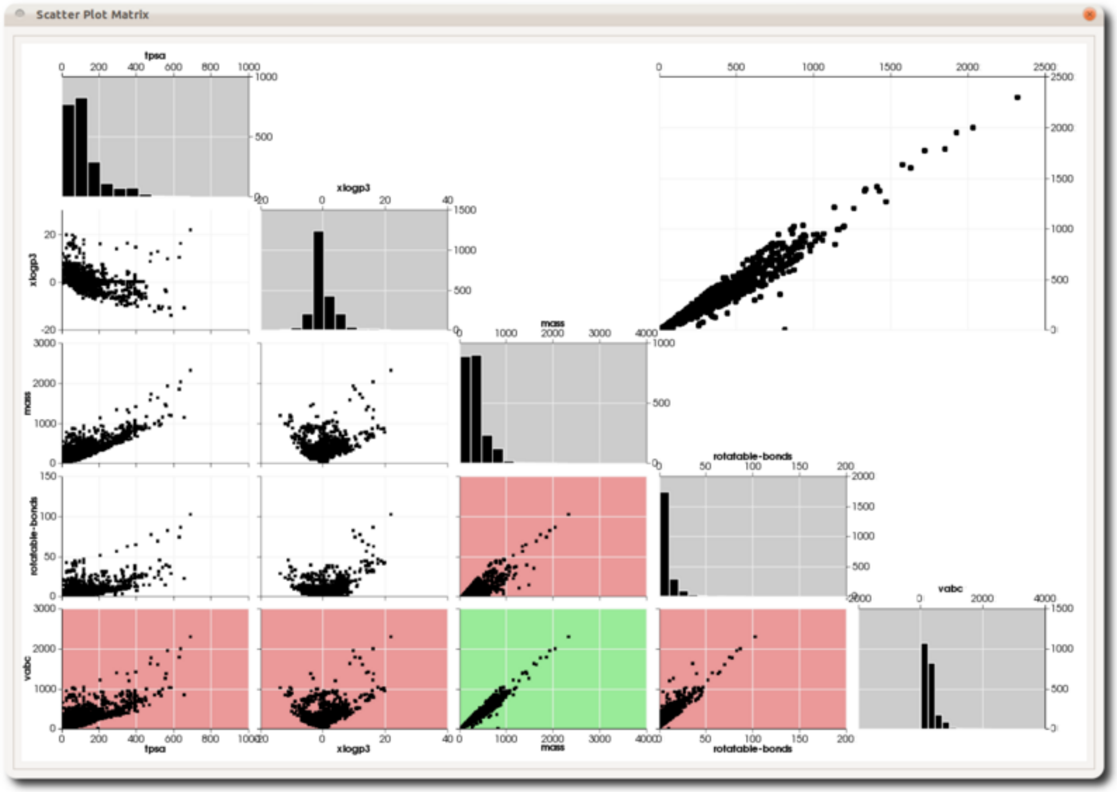


Figure 5: The scatter plot matrix view showing multiple .

The range of charts available has been extended to include simple histograms and scatter plots, through to multivariate visualization techniques such as parallel coordinates and scatter plot matrices (which combine scatter plots for multiple dimensions, along with population histograms for each variable and linked selections, shown in Figure 5). The charts make use of VTK's selection linking functionality that enables users to make and visualize subsets of the data in many different views and representations, as shown in Figure 6.

The use of molecule fingerprinting techniques gives the database the ability to be searched by similarity to a desired structure, as well as enabling queries on chemical name, tag, and other properties. These results can then be viewed in the different data display views available in order further inspect the selected subset of data or do further calculations/export.

Many of the charts in the application feature intelligent tooltips that display the 2D structure along with the IUPAC name, as shown in Figure 7. Network connectivity views

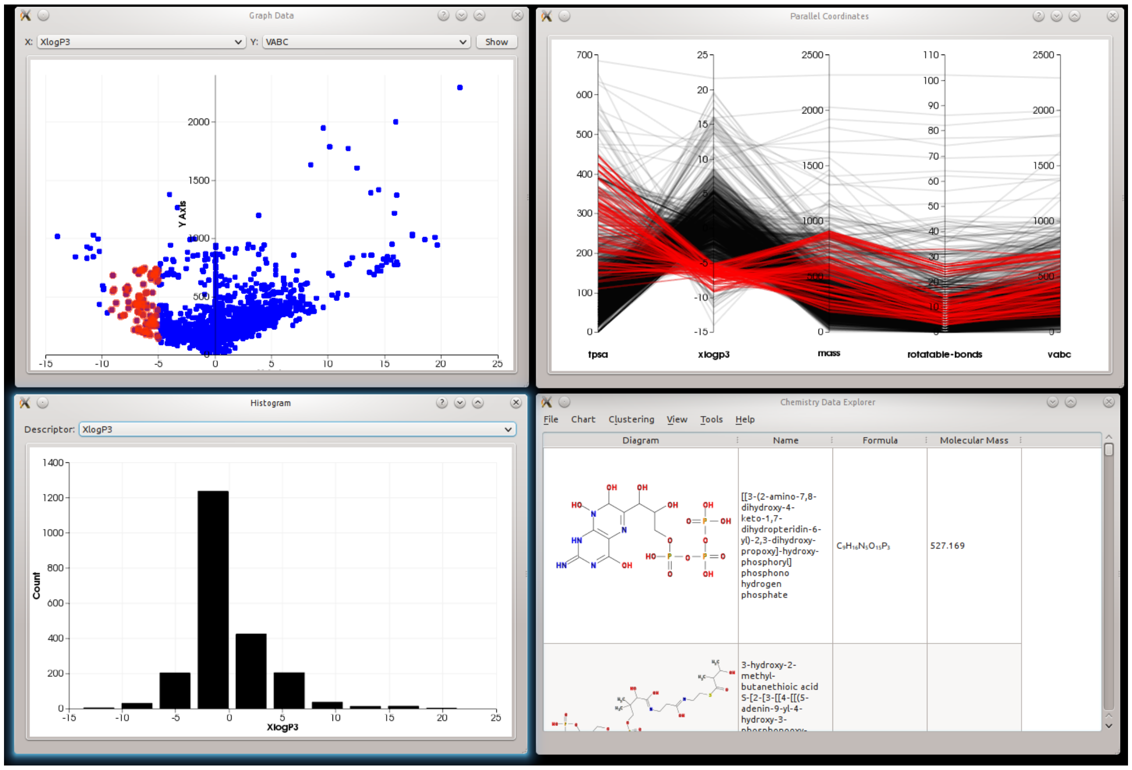


Diagram	Name	Formula	Molecular Mass
	[[3-(2-amino-7,8-dihydroxy-4-keto-1,7-dihydropteridin-6-yl)-2,3-dihydroxy-propoxy]-hydroxy-phosphoryl] phosphono hydrogen phosphate	$C_9H_{16}N_5O_{15}P_3$	527.169
	3-hydroxy-2-methyl-butanethioic acid S-[2-[3-[[4-[[(5-adenin-9-yl-4-hydroxy-3-		

Figure 6: Linked selection in several charts and the table view in ChemData.

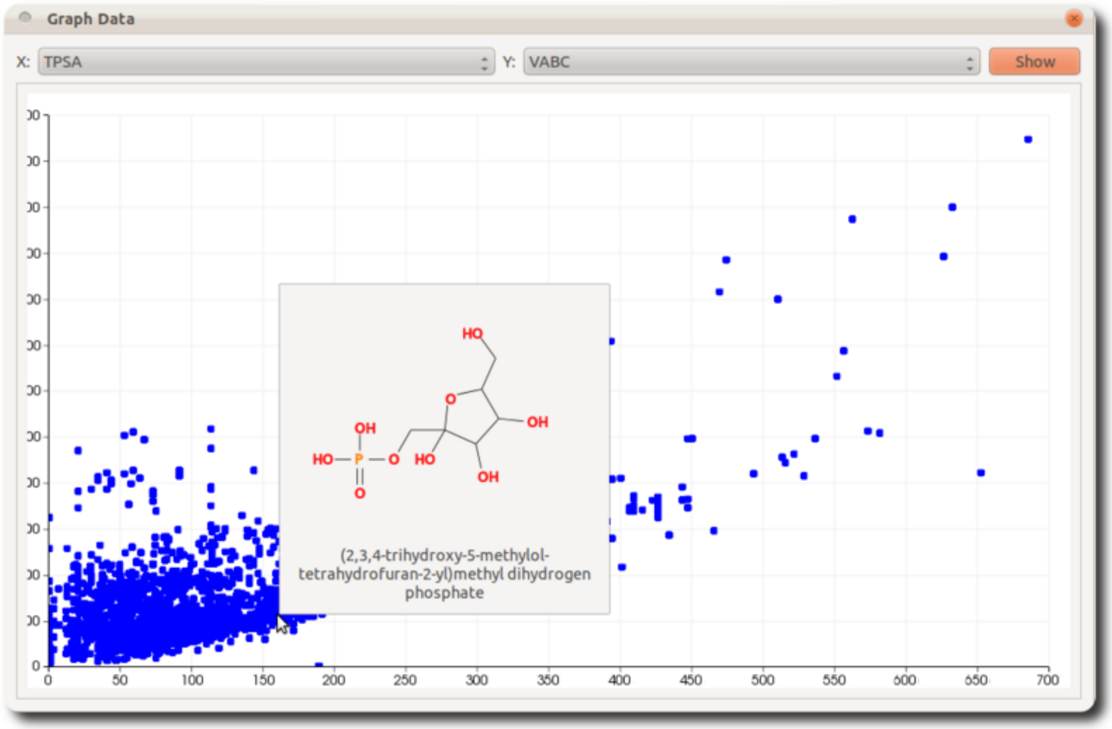
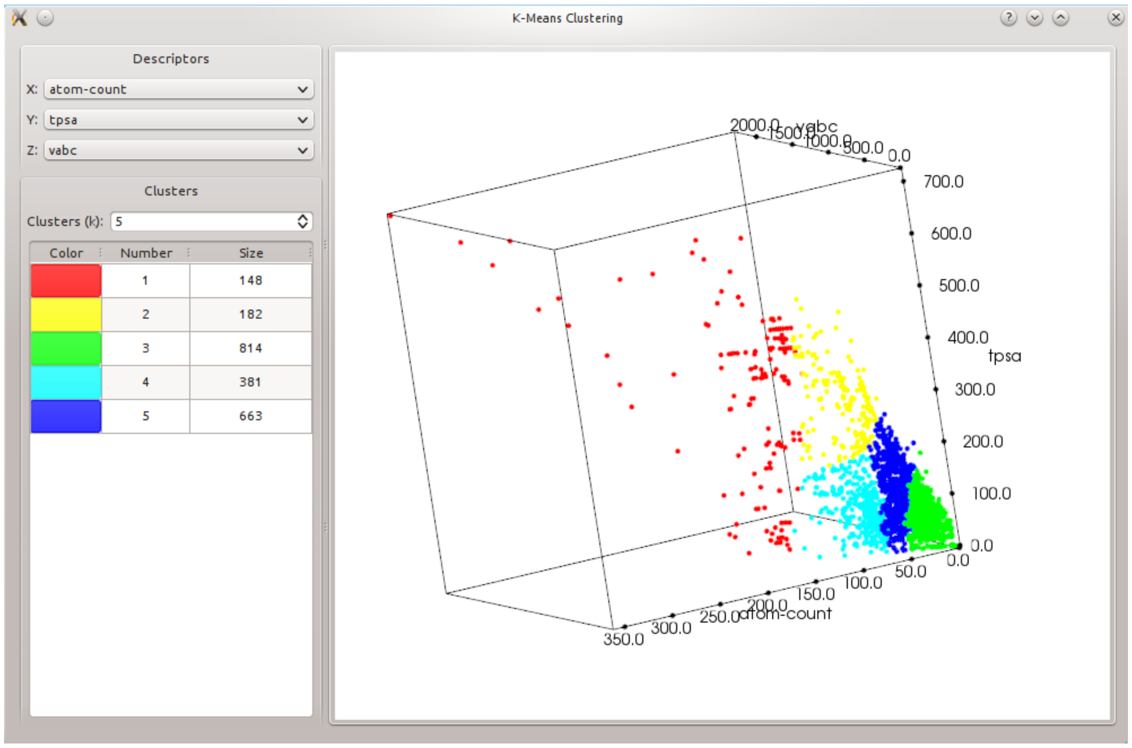

Graph Data
X: TPSA
Y: VABC
Show
(2,3,4-trihydroxy-5-methylol-tetrahydrofuran-2-yl)methyl dihydrogen phosphate

Figure 7: Custom tooltips in scatter plots displaying the 2D structure and IUPAC name of the point under the cursor.

based both on fingerprint and structural similarity enable users to view the overall relatedness of compounds in the database. The K-means clustering view displays descriptor values using 3D charts and groups them based on similarity (See Figure 8); the view features interaction (panning, zooming, etc), and the clustering parameters can be modified.



Descriptors

X: atom-count

Y: tpsa

Z: vabc

Clusters

Clusters (k): 5

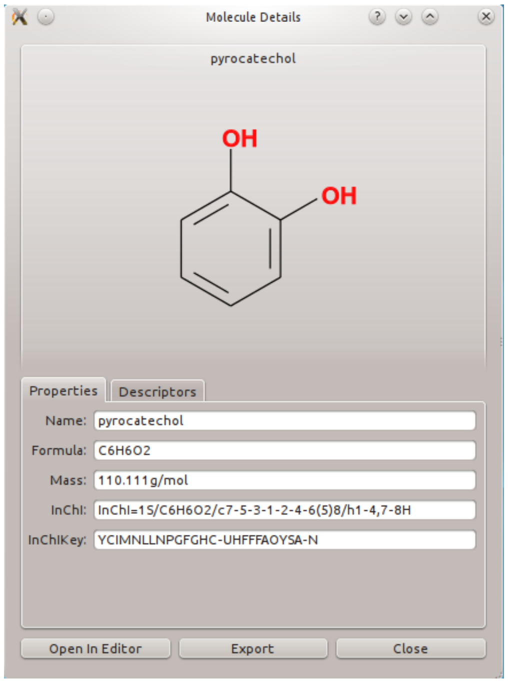
Color	Number	Size
🟥	1	148
🟨	2	182
🟩	3	814
🟦	4	381
🟦	5	663

Figure 8: K-means clustering view in ChemData.

Large data sets can be imported using simple Python scripts, or with graphical tools such as the CSV importer in the application. In addition to the desktop functionality, a prototype web application has been developed which shows the data from the same MongoDB store using modern web techniques coupled with the ParaViewWeb framework to give any user read-only access to the data, and the ability to visualize basic 2D and 3D structure (with interaction for the 3D visualization).

In addition to the charts and table view a detailed dialog is available once a single molecule has been selected, as shown in Figure 9; this gives further details on the structure, such as InChI and SMILES strings. The detailed table views enable the export of structures, or to directly open the structure in Avogadro. Structures can have multiple tags that are searchable, and annotations can be saved with notes relevant to the structure.

A collaboration with the Aspuru-Guzik group at Harvard University has also made available a large number of electronic structure calculations. The data set is over 200TB in size, and an initial 4TB sample has been duplicated for testing and performance benchmarking. The data set is interesting for the ChemData application as it includes a large number of small molecule structures that are candidates for organic solar cell materials,[20] with multiple calculations per structure using different levels of theory and calculation types. A modified version of Q-Chem was used to perform the calculations on the World Community Grid.[21]



Name:	pyrocatechol
Formula:	C6H6O2
Mass:	110.111g/mol
InChI:	InChI=1S/C6H6O2/c7-5-3-1-2-4-6(5)8/h1-4,7-8H
InChIKey:	YCIMNLLNPGFGHC-UHFFFAOYSA-N

Figure 9: The detailed dialog in ChemData for a molecular structure.

2.4 MoleQueue

The MoleQueue application is a C++ Qt application developed to provide an abstraction to local and remote computational resources. Its functionality is not chemistry-specific, but it is necessary in order to remove many of the barriers encountered by users attempting to make use of computational chemistry applications on both the desktop and remote computational resources.

The project provides two components: a system-tray resident application where remote and local computational resources are configured to act as a local job dispatch server, and a client that uses a remote procedure call (RPC) API to stage and submit computational jobs. The RPC API uses a data structure called JSON (JavaScript Object Notation) to carry data in a language and architecture independent fashion. This format was chosen due to the vast array of implementations in virtually every programming language. The JSON RPC 2.0 specification builds upon the JSON data format in order to provide a cross-platform, device-independent RPC API that can easily be implemented in any language desired. Finally, the local socket transport has been chosen due to the security considerations that require local sockets to follow the same permission model as files on every operating system supported - only users with access to the users files on the local system can access a local socket to submit jobs.

It is possible to add further transports, but the communication protocol will remain very simple and easy to implement. A C++ Qt client library is provided, along with some reference implementations for submitting jobs using the Python language. The JSON-RPC 2.0 specification consists of a protocol identifier string-value pair, request, response, notification, and error messages. Requests consist of a JSON-RPC message that contains a method key with a corresponding method name and a params object that contains any method call parameters necessary to complete the request. The client will receive a reply or an error with an id matching that of the request.

Some simple examples, such as requesting a list of queues and their programs is as simple as:

```
{
  "jsonrpc": "2.0",
  "method": "listQueues",
  "id": 42
}
```

The response to this request might look something like this:

```
{
    "jsonrpc": "2.0",
    "result": {
        "Diamond": [
            "GAMESS",
            "MOPAC",
            "Gaussian",
            "NWChem"
        ],
        "Garnet": [
            "GAMESS",
            "MOPAC",
            "Gaussian",
            "NWChem"
        ],
        "Local": [
            "GAMESS",
            "MOPAC",
            "Gaussian",
            "NWChem"
        ]
    },
    "id": 42
}
```

A slightly more complex RPC call to submit a job using MoleQueue would look as follows:

```
{
  "jsonrpc": "2.0",
  "method": "submitJob",
  "params": {
    "queue": "Garnet",
    "program": "GAMESS",
    "description": "B3LYP H2O optimization",
    "inputFile": {
      "filename": "job.inp",
      "contents": "Full contents of input file.\nWill be created in the
         working tree."
    }
  },
  "id": 23
}
```

This submits a job to the remote queue named "Garnet," with the program named "GAMESS". The description is the string that will show up in the MoleQueue user interface, and the input file is specified by either a full path or a file name and contents string. The response for a successful submission looks something like the following:

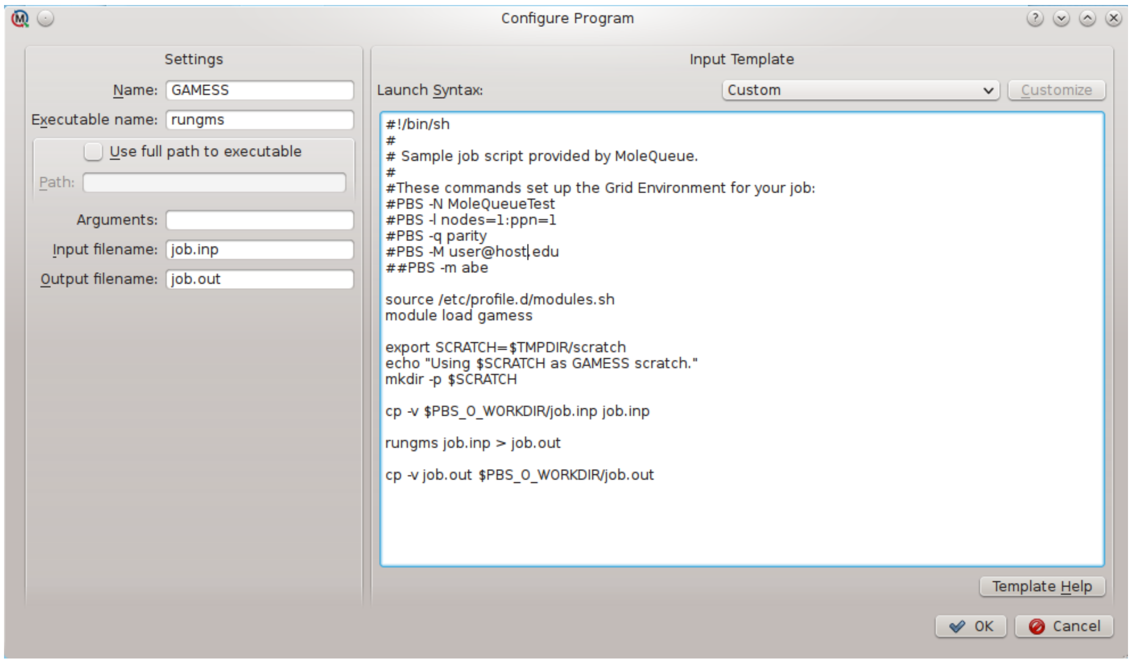
```
{
  "jsonrpc": "2.0",
  "result": {
    "moleQueueId": 17,
    "workingDirectory": "/tmp/MoleQueue/17/"
  },
  "id": 23
}
```

This response object gives a long lived identifier for the job, "moleQueueId," along with the working directory where all files will be staged. If there was an error then an error response will be generated (with an "id" matching that of the request) and an appropriate error, such as queue or program does not exist. Once a job is submitted, notifications are sent when the job state changes; for example, from submitted to running, error, completed etc. Each of the notifications carries the moleQueueId, along with the notification of the job-state change. It is then possible to act upon these changes. There are also RPC methods to query jobs, which can give access to remote job identifiers or to cancel an already submitted job.

The MoleQueue application runs in the system tray, and provides a graphical interface to define remote queues, how to execute programs on those remote queues, and act as an interface for event logging/current job status on all remote systems to which MoleQueue has submitted jobs. Figure 10 shows the program configuration dialog; those familiar with PBS submission scripts should recognize the parameters. A simple keyword substitution is used to replace keywords specified in the template with job-specific settings. This file will be constructed upon job submission and uploaded to the remote system. It will submitted to the batch scheduler, typically using the `qsub` command or its equivalent. Queue/program settings can be imported and exported, facilitating the easy set up of queues across sites if system administrators distribute relevant files with their submission criteria.

In order to be useful on as many high performance computing resources as possible, it was necessary to implement several secure transport methods. The first of which is SSH (secure shell) which is an industry standard employed by many of the world's largest supercomputers to provide access to computational resources. The MoleQueue application can call a specified SSH command line client, or make use of the libssh2 library. The main reason for providing support for both is the lack of Kerberos support in libssh2 and the custom patches applied by some sites to SSH clients in order to support different challenge-response authentication techniques. Once a transport has been chosen to support authentication, command dispatch and file transfer, it is then necessary to support the batch scheduling systems in use on the HPC resource—primarily Sun Grid Engine and PBS.

Support for SSH and with major batch scheduling systems enables MoleQueue to support a large range of supercomputers and cloud resources (including Amazon's EC2 when used with StarCluster to deploy a Sun Grid Engine cluster on demand). The MoleQueue backends are abstracted in such a way to allow for many compute resource backends to be added. In order to take advantage of the HPC resources provided to ERDC researchers, integration with



Name: GAMESS
Executable name: rungms
Use full path to executable
Path:
Arguments:
Input filename: job.inp
Output filename: job.out
Launch Syntax: Custom

```
#!/bin/sh
#
# Sample job script provided by MoleQueue.
#
#These commands set up the Grid Environment for your job:
#PBS -N MoleQueueTest
#PBS -l nodes=1:ppn=1
#PBS -q parity
#PBS -M user@host.edu
##PBS -m abe

source /etc/profile.d/modules.sh
module load gamess

export SCRATCH=$TMPDIR/scratch
echo "Using $SCRATCH as GAMESS scratch."
mkdir -p $SCRATCH

cp -v $PBS_O_WORKDIR/job.inp job.inp

rungms job.inp > job.out

cp -v job.out $PBS_O_WORKDIR/job.out
```

Figure 10: The program configuration dialog in MoleQueue.

the ezHPC platform was also necessary. This has been accomplished using two approaches, the first being the more generic SSH transport coupled with PBS, and the second being the use of the UIT SOAP-like API provided for automated use of HPC resources.

Unfortunately the UIT documentation does not provide enough detail in some instances, and is incorrect in a few places. Only parts of the HPC access have been modeled in SOAP, and so it is necessary to manually parse many of the API responses and match them to the raw PBS response and error codes. Kitware developers have spent significant time using a combination of traditional SOAP implementations, following documentation, trial-and-error, and direct communication with ezHPC UIT support staff in order to provide a working implementation. We have reported several issue/bugs to the UIT support address, and will continue to work with them to resolve these issues. At this stage, the vast majority of the core functionality is now implemented, but there are error conditions and event sequences where the API tokens and data flow remain unclear. Due to the MoleQueue abstraction, it is possible to use a UIT queue or a direct SSH/PBS queue to dispatch, monitor, and retrieve results.

2.5 VTK

The Visualization Toolkit (VTK) is a large, open-source, cross-platform toolkit for data processing and visualization. It has many specialized classes for data processing, informatics, mathematics, data handling, computational fluid dynamics, geospatial visualization, medical imaging, charting, volume rendering, and other areas. One area that has not been added to VTK until now was support for chemical data structures and visualization. Many projects have used VTK for molecular rendering and visualization, but have had to extend it in their own applications and have not been able to benefit from built-in support.

VTK has been extended with a dedicated chemistry module that provides hardware-accelerated visualization making use of advanced support for glyphs in order to get maximum performance. Support for standard atom color schemes and the standard molecular representations have been added. The readers have been augmented to read in secondary protein structure and use the ribbon rendering representations expected for larger biological structures. Additional file format support has been added, along with optimizations for larger structures and interaction.

This means that applications using VTK can benefit from built-in support for chemical structure visualization, along with all the other visualization techniques and data processing code present in the library. The 2D visualization techniques have also been extended in order to better support applications in chemistry, such as custom tooltip support (enabling 2D structures to be displayed in tooltips) and support for multidimensional visualization and selection. Various additional chart types and support for seamless 2D-to-3D chart transitions offer more immersive visualization and analysis environments in Open Chemistry applications. The client-server applications using VTK, such as ParaView and ParaViewWeb, can also benefit from this new functionality and be leveraged in chemical applications.

2.6 Avogadro

The Avogadro project[18] has been developed as a library and desktop application since its inception. Earlier this year, a paper was published discussing the work that went into the project leading up to the 1.0 release series,[19] providing a summary of the development effort. The paper was well received and has increased the profile of the project. After the paper was released the Avogadro 1.1.0 release was made, which incorporates many of the improvements developed during this SBIR project. Development in Phase II of the project has been split between improvements to and stabilization of features in Avogadro 1.1 and rewriting many of the core data structures and algorithms for Avogadro 2, which will move from a GPLv2 license to a more liberal BSD license that allows for much wider use in all sections of government, academia industry, and education.

The Avogadro application is a user-facing application capable of loading, editing and saving chemical structures and loading/analyzing output from many popular computational chemistry codes. It is developed as an open-source community project, with input from across the industry in both research and education. In the latest release of the Avogadro project (1.1.0) significant new features were added, such as support for directly reading GAMESS-US log files among other new codes, automated calculation of all molecular orbital intensities in the background to enable the rapid comparison of orbitals shapes/size, and improved support for vibrational mode animations. A new crystallography extension was added, providing significantly improved support for periodic structures. A crystal library was added to the distribution, along with new builders such as a nanotube builder and chirality inversion. Since its release on September 12, 2012, there have been nearly 20 000 downloads, with the project having over 280 000 recorded downloads of released versions.

The Avogadro 1.1 branch continues to be developed, but development efforts have moved to the 2.0 rewrite (along with porting code). All major contributors agreed to relicense their contributions under the new BSD license, with a new set of libraries being developed to serve the next generation of chemical manipulation and visualization tools. In order to serve everything from desktop applications to HPC client-server deployments, the libraries have been developed using a much more modular pattern. In Avogadro 1 there was a single Avogadro library and an application that linked to it. Avogadro 2 features six libraries at present, and this number will likely grow over the next few months. This is necessitated as heavy computation needs to take place on HPC systems with no graphical environment, whereas the desktop applications needs a range of custom rendering and graphical widgets in order to be user friendly and easy-to-use.

This has been achieved by building a core library that has very few dependencies, avoiding any graphical operations. Data structures, core algorithms, and common definitions are implemented here. A small, focused IO library extends the core library and adds basic file IO, with new dependencies on JSON and XML parsing libraries, and Boost in order to efficiently implement core file format support. The HDF5 library is also needed by the IO library; most other file formats will be translated to these core formats using tools such as Open Babel, Chemkit, and RDKit.

The rendering data structures, support classes, and code reside in a rendering library that depends upon the core library, OpenGL and GLEW, but remains largely independent of the graphical user interface toolkit employed. This opens up the possibility of deploying the rendering code in a wider variety of environments, but requires integration with Qt in order to open up windows, handle user interaction, and perform other common tasks. This is implemented in a Qt OpenGL library that provides customized OpenGL render windows and related functionality. Finally, the desktop widgets necessary for user interaction without OpenGL is in a Qt GUI library, to enable reuse in non-OpenGL applications. Plugin location, loading, and lifetime management are also implemented in the Qt GUI library.

The Avogadro application remains similar to the 1.x version, where a simple Qt application exposes the functionality implemented in the libraries and plugins. This provides an easy entry point for computational chemists, offering customization of the application by changing the loaded plugins and/or writing a custom application. Due to the increased level of modularity deployment in HPC environments of core functionality and/or rendering without bringing along the extra dependencies necessary for the full application paves the way for client-server computing for chemical data processing. The data structures have been designed to be minimal yet functional, and enable scalability from small molecules to extremely large systems.

A fast and lightweight CML reader and writer has been developed using the PugiXML library for XML parsing, and the HDF5 library for storing large amounts of data in a compressed binary format. Significant improvements in load time and memory utilization have been achieved over previous implementations, along with simpler code that can be more easily extended in the future as more features are required. The CML and HDF5 reader/writer has been developed as part of this project, and discussed with experts in the field as an approach to creating standards-compliant and scalable formats for use in chemistry. Converters from other formats are also being developed using two approaches: the extension of Open Babel which already supports a vast array of formats, and the development of JUMBO converters in collaboration with others in the field directly to CML.

Significant advances in the rendering model have added support for vertex buffer objects, vertex and fragment shader programs, and optimized memory layouts for rendering. One of the largest bottlenecks in the rendering pipeline in chemistry is sphere rendering, this has been significantly mitigated through the use of impostor sphere rendering. This approach uses a point sprite or single quad to represent a sphere, the vertex program applies a bill-boarding transform to ensure the quad faces the camera and has the correct dimensions in eye space. The fragment shader then applies lighting equations and an implicit function to update the depth buffer with the correct values to interact with standard rendering techniques. This leads to a highly optimized rendering scene where only four points per sphere need by transformed; and sphere ray-tracing equations can be applied on a per pixel basis, offering not only much improved rendering speed, but pixel perfect sphere boundaries due to the use of an implicit sphere rather than some approximate triangulation method as is traditionally used. The results can be seen in Figure 11.

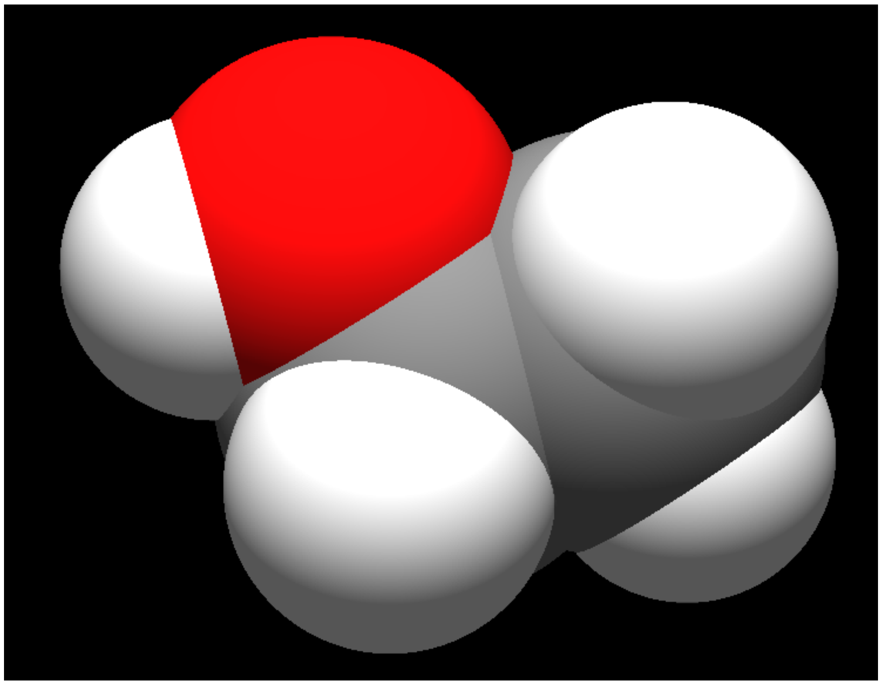


Figure 11: Van der Waals rendering using impostor sphere rendering.

Ideally a similar approach would be used for cylinders, but the gains are lower (cylinders are more complex to model in this way and require less triangles than spheres). Figure 12 shows a typical ball and stick representation, and the seamless joins achieved between the ray-traces spheres and the triangulated cylinders. Through the use of optimized data structures and vastly improved rendering techniques structures containing hundreds of thousands of atoms can now be rendered interactively on commodity laptops when in Avogadro 1.x thousands of atoms would already be displaying performance degradation.

In addition to lower level changes to data structures, rendering and plugins a scalable, client-server oriented architecture is being developed, yielding fast serialization/deserialization of data and simple migration of objects from one process to another (whether local or remote). Integration with MoleQueue and ChemData is in progress.

In addition to the previous approach taken to adding new input generators development of a new methodology is also being explored. Instead of writing a plugin in C++ or Python using the wrapped API from Avogadro calling a script directly from the Avogadro application

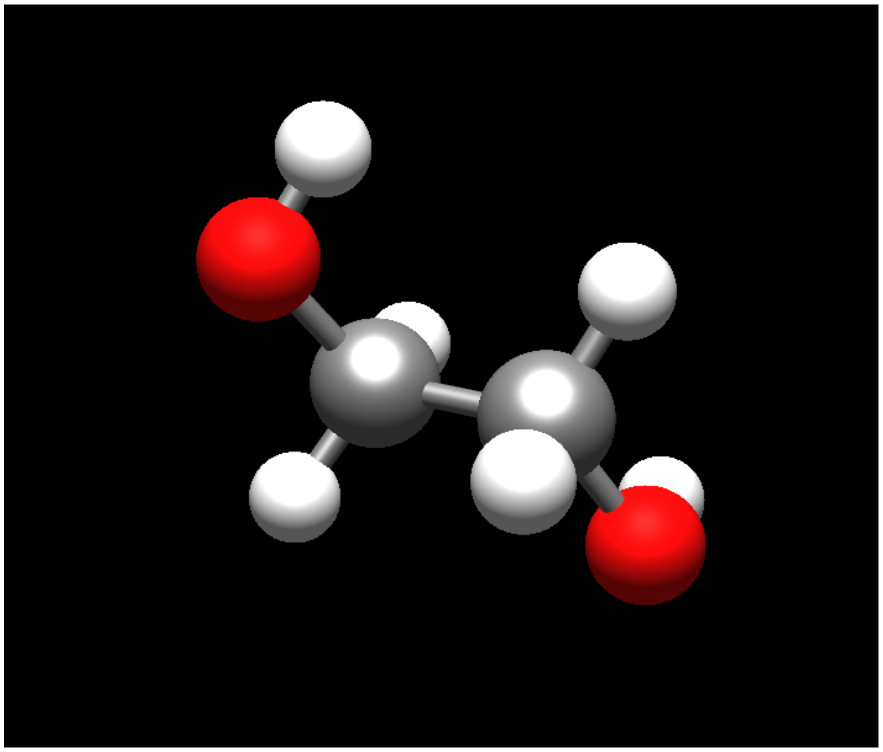


Figure 12: Ball and stick rendering using impostor spheres and triangulated cylinders.

in a separate process is possible. This suffers from a higher startup cost, but benefits greatly from the level of simplicity in designing, adding and editing input generators. All molecular geometry and calculation settings can be passed in using a JSON input to the Python (or any other process) process, and the generated output can be passed back to the application using the standard output of the process. Most languages have support for JSON, and can parse it very efficiently. Calling an independent process removes any issues around consistent linking of the plugin, licensing issues and complexity of learning a new API. This approach can be used in combination with the previous approach, with some input file generators being C++ plugins, and others being implemented in these independent scripts. Should the plugin crash/be unreliable it will not crash the main application process, and the user can be informed.

2.7 OpenQube

The OpenQube project began as part of an Avogadro plugin, and was later split out in order to make it more useful in other applications. It originally contained a minimal molecule data structure, which has since been ported to reuse the structure implemented in Avogadro::Core. Support for several additional output file formats, such as GAMESS, GAMESS-UK, and Molden have been added. Experimental support for CMLComp is also being developed, with a collaboration between NWChem, Open Chemistry, and community members exploring augmenting NWChem and OpenQube with CMLComp support, and developing new converters to go from log file formats to the CMLComp format. The CMLComp convention is being developed in a larger collaboration, with XML dictionaries being developed to extend CML for use in computational chemistry.

Further generalization of OpenQube is also taking place, in order to develop a more widely-applicable and efficient data structure where basis sets can be shared between multiple atoms, support for UHF as well as RHF and higher order Gaussian type orbital functions. This work is nearing the stage where it can be merged to the main code base. Plans are also being made to generalize the parallelization model used in OpenQube; currently only QtConcurrent is supported, which is an efficient map-reduce functional C++ approach. This adds a Qt dependency, which can cause complications with HPC platforms. The current design involves separating out an efficient serial version with re-entrant functions, and adding additional implementations that make use of OpenMP and Intel's thread building blocks library in addition to the existing QtConcurrent implementation. This will add the possibility of client-server molecular orbital and electron density calculations, which are well suited to this approach due to the highly CPU bound nature of the calculations.

Additional testing and verification of results is being added to the library, with a simple command line client to facilitate testing. Output to HDF5 and standard Gaussian cube formats is planned in the near future to enable command line use without any dependency on Open Babel (used for Gaussian cube export thus far).

3 Conclusions

The project is on course to achieve all core goals of the proposed project, and has already prompted several new collaborations that are beginning to see wider impacts in the chemistry community. The three Open Chemistry applications (ChemData, MoleQueue, and Avogadro) are available in both source and binary form for interested early adopters. The JSON-RPC 2.0 inter-process communication APIs and common data structures have been developed to facilitate seamless communication between the loosely coupled components.

The MoleQueue project has already gained some additional funding from an ERDC PETTT project for use in another application domain, and several National Labs have expressed an interest in integrating MoleQueue in their applications, including a climate modeling project demo. This interest has already led to the addition of successful integration with several supercomputers and schedulers (in addition to UIT/ezHPC, Sun Grid Engine and PBS as part of this project). Collaborations with EMSL's NWChem project and Harvard's clean energy project have provided multiple viewpoints on current opportunities in the area for powerful desktop applications in preparing input, integrating with HPC resources and applying cheminformatics techniques to the indexing and analysis of large numbers of computational chemistry result sets.

As the project continues into year 2, further collaborations will be developed in order to gain access to large data sets amenable to client-server visualization and analysis approaches. The project is on track and poised to become a major project at Kitware, opening up new business opportunities in both pre- and post-processing of data in HPC based computational chemistry. Due to the modular implementation, Kitware engineers are exploring novel applications in multi-scale approaches and synergistic activities with other business units. New application areas in materials science and drug discovery are being explored.

References

[1] Qt. Online (June 2010). http://qt.nokia.com/.

[2] Cmake. Online (June 2010). http://www.cmake.org/.

[3] *Mastering CMake*. Kitware, Inc., 5th edition (2010).

[4] Vtk. Online (June 2010). http://www.vtk.org/.

[5] Schroeder, W., Martin, K. and Lorensen, B. *An Object Oriented Approach to 3D Graphics*. Kitware, Inc., 4th edition (2004).

[6] *The VTK User's Guide*. Kitware, Inc., 11th edition (2010).

[7] Open babel. Online (June 2010). http://www.openbabel.org/.

[8] OBoyle, N., Banck, M., James, C., Morley, C., Vandermeersch, T. and Hutchison, G. *Journal of Cheminformatics*, **3**, (2011) 1–14. ISSN 1758-2946. http://dx.doi.org/10.1186/1758-2946-3-33.

[9] Rdkit. Online (June 2010). http://www.rdkit.org/.

[10] Open chemistry. Online (November 2012). http://www.openchemistry.org/.

[11] Code review, topic branches and vtk. Online (April 2012). http://www.kitware.com/source/home/post/62.

[12] Gerrit. Online (November 2012). http://code.google.com/p/gerrit/.

[13] Cdash@home. Online (October 2010). http://www.kitware.com/source/home/post/21.

[14] Json specification. Online (November 2012). http://www.json.org/.

[15] Mongodb. Online (November 2012). http://www.mongodb.org/.

[16] Bson specification. Online (November 2012). http://bsonspec.org/.

[17] Murray-Rust, P. and Rzepa, H. S. *Chemistry Intl.*, **4**(24), (2002) 9–13.

[18] Avogadro. Online (June 2010). http://avogadro.openmolecules.net/.

[19] Hanwell, M. D., Curtis, D. E., Lonie, D. C., Vandermeersch, T., Zurek, E. and Hutchison, G. R. *Journal of Cheminformatics*, **4**(1), (2012) 17.

[20] The clean energy project. Online (November 2012). http://cleanenergy.harvard.edu/.

[21] World community grid. Online (November 2012). http://www.worldcommunitygrid.org/.